Electronic Version

Stylesheet Version v1.1.1

Description

Post-Inverse-Quantization AC Prediction with a Unified Variable-Length-Decoding and Inverse-Quantization Stage

BACKGROUND OF INVENTION

[0001] This invention relates to video compression, and more particularly to decoder pipelines having coefficient prediction.

[0002] One fast-growing area of development is video transmission and compression. Video data can greatly enhance the quality of a computing experience. The consumer use of the Internet took off once graphics was linked to earlier text-based web pages. Portable consumer devices such as cell phones and personal digital assistant (PDA's) are being equipped with small cameras to allow for capture of still or even video pictures. Efficient transmission of captured images over limited-bandwidth links requires some

sort of compression of the images.

[0003] A number of video-compression techniques are known. Compression standards, such as those developed by the motion-picture-experts group (MPEG), have been widely adopted. These compression techniques are lossy techniques, since some of the picture information is discarded to increase the compression ratio. However, compression ratios of 99% or more have been achieved with minimal noticeable picture degradation.

[0004] Next-generation compression standards have been developed for transmitting video over wireless networks. The MPEG-4 standard provides a robust compression technique for transmission over wireless networks. Recovery can occur when parts of the MPEG-4 bit stream is corrupted.

[0005] These MPEG standards ultimately break the image up into small 16x16 pixel macroblocks or even smaller 8x8 pixel blocks. Each block can then be compressed more or less independently of other blocks, and movement of blocks can be described as highly compressed "motion vectors" rather than large bitmaps of pixels.

[0006] Figure 1 shows an image frame divided into rows and columns of blocks. The MPEG standard uses a divide-

and-conquer technique in which the video sequence is divided into individual image frames known as video object planes (VOPs), and each frame is divided into rows and columns of macroblocks. Each macroblock is a rectangle of 16 by 16 pixels. Each macroblock can be further divided into 8x8 blocks.

[0007] Various window sizes and image resolutions can be supported by MPEG standards. For example, one common format is an image frame of 176 by 144 pixels. The image frame is divided into 18 rows of 8x8 blocks, with each row having 22 blocks each of 8x8 pixels. A total of 396 blocks are contained in each frame.

[0008] The blocks are arranged in a predetermined order, starting in the upper left with the first block (BLK #0). The second block, BLK #1, is to the right of BLK #0 in the first row, followed by blocks #2 to BLK #21 in the first row. The second row contains BLK #22 to BLK #43. The last row contains BLK #374 to BLK #395. Of course, other image sizes and formats can have the blocks in rows of various lengths, and various numbers of rows.

[0009] When an image frame is encoded, each block is encoded in macroblock-order, starting with the first macroblock of four blocks BLK #0,1 in the first row, and BLK #22, 23 in

the second row, continuing to BLK #20, 21 in the first row and BLK# 42, 43 in the second row, , and on until the last two rows to BLK #395. The blocks are arranged in the bit stream into one or more video packets (VP) with a header. In this example Y values of pixels are shown.

[0010] Figure 2A shows an 8x8 block of pixels. Pixels can have components, such as luminance Y values and U, V chrominance values. An 8x8 pixel block may have 64 Y values but only half or one-quarter as many U or V values. Figure 2A shows an 8x8 array of pixel values for block N. Each pixel is identified by its block number, column, and row. Thus pixel P(55,2,6) is the pixel at the second column of the sixth row of block 55. The 8x8 blocks can be Y values or U or V values, but only half as many rows and columns exist for U and V values for some common YUV formats.

[0011] Figure 2B shows an 8x8 block of quantized DCT coefficients. After motion compensation blocks are compressed. During compression, the block of 8x8 pixel values shown in Fig. 2A is transformed from the spatial into the frequency domain using a transform such as a discrete cosine transform (DCT). The result of the DCT is an 8x8 array of coefficients. The DCT coefficients are desig-

nated D(b,c,r), where b is the block #, c is the column #, and r is the row #.

[0012] The DCT coefficients D(b,c,r) are then scaled or quantized by a quantization parameter Q. For example, when Q is 6, DCT coefficient values between 0 and 5 can be scaled to 0, while values between 6 and 11 are scaled to 6. These scaled values can be level-shifted by half the scale: 0 is shifted to 3, while 6 is shifted to 9. Since 12 input values (0-11) are reduced to 2 output value (3,9), compression occurs during quantization. Quantization changes the DCT coefficients D(b,c,r) into quantized DCT coefficients E(b,c,r). The quantized DCT coefficients E(b,c,r) are sometimes referred to as AC coefficients, except for the first coefficient E(b,0,0), which is known as the DC coefficient.

[0013] Although there are the same number and arrangement of quantized coefficients as pixels, there is not a one-to-one correspondence of coefficient values to pixel values. Pixel P(12,4,3) does not directly correspond to coefficient E(12,4, 3), even though they have the same position in block 12's array. A pixel cannot be generated from only the coefficient at the same position. Instead, many coefficients are needed to re-generate a pixel value using an inverse DCT.

[0014] In general, coefficients in one block do not have much correlation to coefficients in other blocks. However, the first row and column of coefficients does show some correlation to other blocks. The very first (DC) coefficient E(N,0,0) is unique to each block. However, other "alternating-current" (AC) coefficients in the first row and first column show correlation to coefficients in other blocks. These first-row and first-column coefficients are known as A.C. coefficients, and can be predicted using a technique known as adaptive AC coefficient prediction. D.C. coefficients can be predicted by another mechanism.

[0015] Figure 3A shows correlation of first-row AC coefficients with the first row of the block above. A current block N has a first row of DCT coefficients E(N,0,0), E(N,1,0), E(N,2,0), E(N,3,0)...E(N,7,0). The first coefficient is E(N,0,0). The remaining coefficients E(N,1,0), E(N,2,0), E(N,3,0)...E(N,7,0) in block N often show a correlation to the first-row coefficients in the block above, block N-L, where L is the number of rows (22 in this example). Thus the first-row AC coefficients E(N,1,0), E(N,2,0), E(N,3,0)...E(N,7,0) in block N often show some correlation to the first-row AC coefficients E(N-22,1,0), E(N-22,2,0), E(N-22,3,0)...E(N-22,7,0) in the above block N-22.

[0016] The first-row AC coefficients E(N,1,0), E(N,2,0), E(N,3,0)...E(N,7,0) in block N can be differentially coded as differences from the first-row AC coefficients E(N-22,1,0), E(N-22,2,0), E(N-22,3,0)...E(N-22,7,0) in block N-L, improving compression since the difference values require fewer bits than do full coefficient values. During decompression, block N's first row coefficients E(N,1,0), E(N,2,0), E(N,3,0)...E(N,7,0) can be generated from E(N-22,1,0), E(N-22,2,0), E(N-22,3,0)...E(N-22,7,0).

[0017] Figure 3B shows correlation of first-column AC coefficients with the first column of the previous block to the left. A current block N has a first column of DCT coefficients E(N,0,0), E(N,0,1), E(N,0,2), E(N,0,3)...E(N,0,7). The first coefficient E(N,0,0) is predicted using another mechanism, but the remaining coefficients E(N,0,1), E(N,0,2), E(N,0,3)...E(N,0,7) in block N often show a correlation to the first-column coefficients in the prior above, block N-1. Thus the first-column AC coefficients E(N,0,1), E(N,0,2), E(N,0,3)...E(N,0,7) in block N often show some correlation to the first-column AC coefficients E(N-1,0,1), E(N-1,0,2), E(N-1,0,3)...E(N-1,0,7) in the prior block N-1. The first-column coefficients for block N can be coded as differences with the first-column coefficients in block N-1 and

later re-generated during de-compression using AC prediction.

[0018] Figure 4 is a block diagram of a prior-art decoder that performs AC prediction between the variable-length decoder and the inverse-quantizer. Parser 20 receives an MPEG-compressed bit-stream and extracts an AC prediction flag for the current block. When the AC prediction flag is false, no AC prediction is performed. Instead, mux 19 selects the decoded G(j,k) output from variable-length decoder 10. Variable-length decoder 10 receives the 8x8 block's coefficients from parser 20 and decodes the block's portion of the bit-stream using a variable-length decoding. For example, common coefficient values can be coded with a few bits while uncommon coefficient values can be coded as longer bit codes.

[0019] The decoded G(j,k) coefficients selected by mux 19 are the quantized DCT coefficients E(j,k), where j is the column and k is the row for the coefficient in the current block N. Inverse-quantizer 14 performs the inverse of the scaling and quantization step performed by the encoder, using the quantization parameter Qe that parser 20 extracts from the bit-stream. The quantization parameter Qe can be different for different blocks, or can be the

same for all blocks in a frame. Inverse-quantizer 14 generates the reconstructed DCT coefficients D(j,k). Inverse DCT 16 performs an inverse DCT operation on the on all coefficients in the 8x8 block at one time. D(j,k) coefficients to generate the reconstructed pixel values P(j,k) for the block. The pixel values can be stored or displayed.

[0020] When the AC prediction flag is true, AC prediction is performed. Mux 19 selects the predicted quantized DCT coefficients GP(j,k) from the lower input rather than the bit-stream quantized DCT coefficients G(j,k) from variable-length decoder 10. Adder 22 generates the predicted quantized DCT coefficients GP(j,k) by adding an adjusted stored coefficient to the bit-stream quantized DCT coefficients G(j,k), which contain a differential value.

[0021] Quantized DCT coefficients from prior blocks are stored by coefficient store 18. The quantized DCT coefficients E(j,k) for the first row and for the first column of each current block are copied to coefficient store 18. Later, these stored coefficients can be read from coefficient store 18 as the prior block (N-1) or above block's (N-L) stored coefficients. The prior blocks' quantization parameters Qf are also stored.

[0022] The stored coefficients from coefficient store 18 must be

adjusted since the quantization parameters for the current and prior blocks may be different. The stored coefficients for the prior block (either block above N-L or block to the left N-1) is read from coefficient store 18 and multiplied by the stored quantization parameter Qf for the prior block by multiplier 26. The result of multiplier 26 is then divided by the current block's quantization parameter Qe using divider 24. Divider 24 can be an integer divider that rounds to the nearest integer. Half-integer values are rounded away from zero.

[0023] Multiplier 26 and divider 24 thus correct the stored coefficient from coefficient store 18 for the differing quantization parameters. Multiplier 26 and divider 24 adjust the stored coefficients by a ratio of Qf to Qe. The adjusted stored coefficient is then added by adder 22 to the bitstream quantized DCT coefficients G(j,k), generating the predicted quantized DCT coefficients GP(j,k). Mux 19 selects the predicted quantized DCT coefficients for the first row or first column and sends these to inverse-quantizer 14.

[0024] The overall effect of AC prediction is to generate the reconstructed quantized DCT coefficients for the first row or column:

[0025] $E = G + (F * Qf)//Qe,$

[0026] where F is the stored coefficient from coefficient store 18 and // represents integer division with integer rounding away from zero. Other rows and columns are passed through the upper input of mux 19:

[0027] $E = G$

[0028] While such an AC predictor is useful, such AC prediction is performed between variable-length decoder 10 and inverse-quantizer 14. This precludes a simple unified variable-length decoder and inverse-quantizer. What is desired is a unified variable-length decoder and inverse-quantizer stage. A unified variable-length decoder and inverse-quantizer is desired that still allows for AC prediction.

Brief Description of Drawings

[0029] Figure 1 shows an image frame divided into rows and columns of blocks.

[0030] Figure 2A shows an 8x8 block of pixels.

[0031] Figure 2B shows an 8x8 block of quantized DCT coefficients.

[0032] Figure 3A shows correlation of first-row AC coefficients with the first row of the block above.

[0033] Figure 3B shows correlation of first-column AC coefficients with the first column of the previous block to the left.

[0034] Figure 4 is a block diagram of a prior-art decoder that performs AC prediction between the variable-length decoder and the inverse-quantizer.

[0035] Figure 5 is an overview of AC prediction after a unified variable-length decoder and inverse-quantizer stage.

[0036] Figure 6 shows in more detail an AC predictor that operates after inverse-quantization.

[0037] Figure 7 shows a Q-subtractor block.

[0038] Figure 8 shows coefficients and quantization parameters stored in the coefficient store.

DETAILED DESCRIPTION

[0039] The present invention relates to an improvement in video de-compression. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present

invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0040] MPEG bit-streams may be coded and compressed by a variety of coders and typically follow the MPEG specification. AC prediction is expected to be performed on the quantized DCT coefficients before inverse-quantization.

[0041] The inventor has realized that a unified, single-stage variable-length decoder and inverse-quantizer is desirable. A dedicated, unified variable-length decoder and inverse-quantizer could allow both to be performed in a single step or clock cycle, improving decoder performance. Since AC prediction is performed only on the first row or column, it can be performed later, after the inverse-quantizer. The inverse-quantizer performs the operation:

[0042] $D = IQ(E,Qe) = 2*E*Qe + sign(E)*odd(Qe)$

[0043] where sign(E) is -1, 0, +1 when E is negative, zero, or positive, and odd(Qe) is Qe for odd numbers, Qe-1 for even numbers.

[0044] The inventor realizes that AC prediction can be moved to occur after the inverse-quantizer, allowing the inverse-quantizer to be combined with the variable-length de-

coder. The inventor performs AC prediction on the reconstructed DCT coefficients D(j,k) output by the inverse-quantizer, rather than the quantized DCT coefficients E(j,k) output by the variable-length decoder.

[0045] Figure 5 is an overview of AC prediction after a unified variable-length decoder and inverse-quantizer stage. Unified VLD/IQ stage 15 includes the functions of variable-length decoder 10 and inverse-quantizer 14 in a single stage. Unified VLD/IQ stage 15 receives a coded 8x8 block from parser 20 and generates DCT coefficients C(j,k). Intermediate quantized DCT coefficients G(j,k) may not be directly generated by unified VLD/IQ stage 15.

[0046] When parser 20 reads the AC prediction flag for the current block as false, mux 56 outputs C(j,k) from unified VLD/IQ stage 15 as DCT coefficients D(j,k). Inverse DCT 16 then generates pixels P(j,k) for the block from DCT coefficients D(j,k).

[0047] When the AC prediction flag is true, mux 56 selects predicted DCT coefficients from calculator 60 for the first row or column. Calculator 60 receives the current block's quantization parameter Qe from parser 20. Qe is also sent to inverse-quantizer 14. Calculator 60 also reads the prior block's quantization parameter Qf and the prior block's

DCT coefficients D(j,k) that were earlier stored by coefficient store 58 for the previous block N-1 or the block above N-L.

[0048] Calculator 60 performs a more complex calculation than for AC prediction of Fig. 4. However, calculator 60 is only needed for one row or one column of a block, and only when the block has its AC prediction flag set. The performance improvement of unified VLD/IQ stage 15, which improves performance for all coefficients in all blocks, more than makes up for the complexity of calculator 60.

[0049] Rather than store the quantized DCT coefficients E(j,k), coefficient store 58 stores the DCT coefficients D(j,k) after inverse-quantization.

[0050] Figure 6 shows in more detail an AC predictor that operates after inverse-quantization. Inverse-quantizer 14 of unified VLD/IQ stage 15 outputs reconstructed DCT coefficients C(j,k) after scaling G(j,k) by the current quantization parameter Qe. Mux 56 selects C(j,k) when the AC prediction flag is not set, or for non-first rows and columns. These coefficients C(j,k) become the DCT coefficients D(j,k) applied to inverse DCT 16 to generate the block's pixels P(j,k).

[0051] The reconstructed DCT coefficients D(j,k) and quantiza-

tion parameter Qe are stored in coefficient store 58 for the first row and first column of each block. Later, these stored coefficients can be used for AC coefficient prediction by calculator 60.

[0052] The inventor defines a subtract-Q "SubQ" function as:

[0053] $SubQ(x,Q) = (x - sign(x)*odd(Q))/2$,

[0054] where sign(x) is -1, 0, +1 when x is negative, zero, or positive, respectively. Odd(Q) is Q when Q is odd, or Q-1 when Q is even.

[0055] When x is positive and Q is odd, SubQ is half of the difference between x and Q, and is positive when x is larger than Q. When x is negative and Q is odd, SubQ is still half of the difference between x and Q, but is negative when |x| is larger than Q. The quantization parameter Q is always positive, but x, which is a coefficient, may be negative or positive. When Q is even, it is rounded down to the next odd integer, so SubQ is not exactly half the difference for even Q.

[0056] Q-subtractor 68 receives the stored coefficients B(j,k) from coefficient store 58 as the x input, and the stored quantization parameter Qf as the Q input, and outputs the SubQ function result to divider 84. The SubQ result for the

stored coefficient is divided by the current quantization parameter Qe by divider 84 and then multiplied by the current quantization parameter Qe with multiplier 82. Since divider 84 is an integer divider that rounds to the nearest integer, the combination of multiplying and dividing by the same number (Qe) is not necessarily 1. Thus multiplier 82 and divider 84 correct stored coefficients for integer-division rounding.

[0057] Q-subtractor 64 performs the SubQ function on the current-block DCT coefficients C(j,k), using the current block's quantization parameter Qe. The SubQ result is added to the corrected stored coefficient from multiplier 82 by adder 72.

[0058] The sum from adder 72 is doubled by multiplier 74 to generate coefficient value X. A final adjustment is made to X to generate the predicted DCT coefficients CP(j,k) input to mux 56. The current quantization parameter Qe is rounded down to the nearest odd integer by odd generator 62. Then the sign of X is applied to the odd Qe by multiplier 78, which does not have to be a full multiplier. The signed, odd Qe is then added to coefficient X by final adder 76 to generate predicted DCT coefficients CP(j,k) input to mux 56, than become the reconstructed DCT co-

efficients D(j,k) when AC prediction is performed.

[0059] Figure 7 shows a Q-subtractor block. Odd rounder 94 rounds down to the nearest odd integer of input Q. The sign of input P is multiplied by sign multiplier 96 with the output of odd rounder 94 and the result is input to subtractor 92. Subtractor 92 subtracts the result of sign multiplier 96 from input P. Divider 93 divides the output of subtractor 92 by 2 to generate the SubQ(P,Q) result of Q-subtractor 68. The SubQ function is defined as SubQ(P,Q) = (P - sign(P)*odd(Q))/2.

[0060] Figure 8 shows coefficients and quantization parameters stored in the coefficient store. For each block N, the reconstructed DCT coefficients for the first row D(N,j,0) and first column D(N,0,k) are stored in coefficient store 58. The quantization parameter Qe for that block is also stored as Q(N). As new blocks are processed, the current-block pointer N is advanced.

[0061] When AC prediction is performed to the prior block, the first column coefficients D(N-1,0,k) for the prior block are read from coefficient store 58 as the stored coefficients. The stored Q(N-1) is also read as the stored quantization parameter Qf. When AC prediction is performed to the block above, the first row coefficients D(N-L,j,0) for the

above block N-L, where L is the number of blocks in a row, are read from coefficient store 58 as the stored coefficients. The stored Q(N-L) is also read as the stored quantization parameter Qf.

[0062] Equations for Functions Performed

[0063] The functions performed by the blocks of calculator 60 in Fig. 6 can be derived as follows:

[0064] The quantization parameter Q is always a positive number. B(j,k) represents the stored coefficients D(j,k) for the prior block N-1 or N-L, either for the first row or first column. Qf is the stored quantization parameter for block N-1 of N-L.

[0065] The subtract-Q or "SubQ" function is defined as:

[0066] SubQ(x,Q) = (x - sign(x)*odd(Q))/2,

[0067] where sign(x) is -1, 0, +1 when x is negative, zero, or positive, respectively.

[0068] The odd function Odd(Q) is Q when Q is odd, or Q-1 when Q is even.

[0069] AC Prediction can be defined as:

[0070] E = G + (F*Qf)//Qe (Eqn. 1)

[0071] Inverse-quantization can be defined as:

[0072] $D = IQ(E,Qe) = 2*E*Qe + sign(E)*odd(Qe)$ (Eqn. 2)

[0073] If the inverse-quantization is performed on the VLD output G rather than E to generate C:

[0074] $C = IQ(G,Qe) = 2*G*Qe + sign(G)*odd(Qe)$

[0075] Re-arranging:

[0076] $2*G*Qe = C - sign(G)*odd(Qe)$

[0077] Since the sign(C) = sign(G):

[0078] $2*G*Qe = C - sign(C)*odd(Qe)$

[0079] Substituting the SubQ function:

[0080] $G*Qe = (C - sign(C)*odd(Qe))/2 = SubQ(C,Qe)$ (Eqn. 3)

[0081] The prior-art stored quantized DCT coefficients E as the predictor F. In the invention, DCT coefficients D after the inverse-quantizer are stored as the predictor B. The new B is the inverse-quantization of the old F:

[0082] $B = IQ(F,Qf) = 2*F*Qf + sign(F)*odd(Qf)$

[0083] Re-arranging:

[0084] $2*F*Qf = B - sign(F)*odd(Qf)$

[0085] Since the sign(B) = sign(F):

[0086] $F*Qf = (B - sign(B)*odd(Qf))/2$

[0087] Substituting the SubQ function:

[0088] F*Qf = (B - sign(B)*odd(Qf))/2 = SubQ(B,Qf) (Eqn. 4)

[0089] AC Prediction can be defined as:

[0090] E = G + (F*Qf)//Qe (Eqn. 1)

[0091] Inverse-quantization can be defined as:

[0092] D = IQ(E,Qe) = 2*E*Qe + sign(E)*odd(Qe) (Eqn. 2)

[0093] Substituting for E using Eqn. 1:

[0094] D = IQ(E,Qe) = 2*G*Qe + 2*((F*Qf)//Qe)*Qe + sign(E)*odd(Qe)

[0095] Substituting with Eqn. 3, where G*Qe = SubQ(C,Qe):

[0096] D = 2*SubQ(C,Qe) + 2*((F*Qf)//Qe)*Qe + sign(E)*odd(Qe)

[0097] Further substituting with Eqn. 4, where F*Qf = SubQ(B,Qf):

[0098] D = 2*SubQ(C,Qe) + 2*(SubQ(B,Qf)//Qe)*Qe + sign(E)*odd(Qe)

[0099] Define X as 2*SubQ(C,Qe) + 2*(SubQ(B,Qf)//Qe)*Qe and substitute:

[0100] D = X + sign(E)*odd(Qe)

[0101] Since sign(D) = sign(E) = sign(X):

[0102] D = X + sign(X)*odd(Qe).

[0103] X is the output of multiplier 74, which is the value of node X between 74 and 76 in Fig. 6.

[0104] Of course, when no AC prediction is done,

[0105] D = C

[0106] ALTERNATE EMBODIMENTS

[0107] Several other embodiments are contemplated by the inventor. A more complex AC prediction flag may be used to select whether the first row or the first column is predicted. Numbers such as the quantization parameters could be shifted, inverted, etc.

[0108] The functional and computational blocks can be implemented in a variety of ways, such as by firmware routines in a digital-signal processor (DSP) chip, or in logic in a logic array chip, or as software routines executed by a processor, or a combination of techniques. The blocks can be partitioned in many different ways. A programmable register can allow calculations to be disabled, or allow for different threshold values or equations to be used. Additional dividers, multipliers, inverters, etc. could be added to achieve the same or similar results. Active-low rather than active-high signals may be used, and various encodings can be substituted.

[0109] Other video formats, frame sizes, and block sizes could be supported. Many other functional blocks can exist in a complex MPEG decoder, and pipelining logic and staging registers may also be present. Various pipelining registers can be added. Different versions of the MPEG or other compression standards could be supported.

[0110] The abstract of the disclosure is provided to comply with the rules requiring an abstract, which will allow a searcher to quickly ascertain the subject matter of the technical disclosure of any patent issued from this disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. 37 C.F.R. § 1.72(b). Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC § 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not only the structures described herein for performing the function and their

structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures since they both perform the function of fastening. Claims that do not use the word means are not intended to fall under 35 USC § 112, paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0111] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.